DATA	ZMIANY
24.01.2025	*Wojciech Midura (wmidura@student.agh.edu.pl)*

BAZA DANYCH Z INTERFEJSEM GRAFICZNYM

Autor: Wojciech Midura
Akademia Górniczo-Hutnicza

Kraków 2025

Spis treści

Lista oznaczeń

1. Wstęp

Tworzona baza danych ma na celu przechowywanie podstawowych informacji o filmach, takich jak: rok produkcji, tytuł, imię i nazwisko reżysera oraz gatunek filmowy. Aplikacja umożliwia użytkownikowi łatwe znalezienie interesującego go filmu, na przykład na podstawie tytułu lub gatunku.

Dzięki przystępnemu interfejsowi użytkownik może w prosty sposób przeglądać informacje zawarte w bazie danych. Funkcje wyszukiwania i sortowania pozwalają na szybkie odnalezienie filmu, który odpowiada jego preferencjom. Aplikacja oferuje również możliwość dodawania nowych rekordów oraz edytowania istniejących danych, co jest szczególnie przydatne w sytuacjach, gdy film znajdzie się na liście „do obejrzenia" lub gdy konieczna jest korekta błędnych informacji. Użytkownik ma także możliwość usuwania filmów z bazy danych.

Jednym z kluczowych elementów projektu jest funkcjonalność zapisywania wszystkich rekordów do pliku oraz ich odczytywania. Dzięki temu można tworzyć różnorodne bazy danych, takie jak „Ulubione Filmy", „Filmy do Obejrzenia" czy „Filmy do Polecenia". Umożliwia to także łatwe udostępnianie list filmów innym użytkownikom.

Projekt bazuje na wykorzystaniu bibliotek Qt oraz klas takich jak: QTableWidget, QJsonArray, QJsonObject i QJsonDocument. Główne okno programu zawiera tabelę do wyświetlania bazy danych oraz zestaw przycisków, które umożliwiają nawigację. Funkcje takie jak „Add" i „Modify" otwierają dodatkowe okna, pozwalające na dodawanie lub modyfikację informacji o filmach.

Wymagania systemowe:

Aby możliwe było zbudowanie projektu, należy zainstalować biblioteki Qt oraz sklonować repozytoriom GitHub wraz z submodułami.

Biblioteki Qt są dostępne na stronie: Qt Installer. Podczas instalacji, w sekcji „Select Components", należy otworzyć zakładkę „Qt" i wybrać następujące pakiety rozszerzeń:

MinGW 13.1.0 64-bit (instaluje środowisko IDE Qt),

MSVC 2022 64-bit (umożliwia integrację bibliotek Qt z Visual Studio 2022).

VisualStudio 2022:

Szczegółowy proces instalacji rozszerzeń został przedstawiony w filmie: Instalacja Qt dla Visual Studio 2022 [8]. Po zakończeniu konfiguracji należy otworzyć **Plik->Otwórz->CMake..** wskazać CMakeLists.txt i zbudować aplikację.

Qt Creator:

W przypadku Qt IDE również należy otworzyć CMakeLists.txt.

2. Funkcjonalność

Podstawową funkcjonalnością bazy danych jest przechowywanie inforamcji w niej zawartych, umożliwienie ich odczytu i modyfikacji.

Główne okno projektu posiada przyciski do nawigacji takie jak:

- „Add” – otwiera on nowe okno w którym użytkownik może wprowadzać dane nowego rekordu, sprawdzane jest tam czy wszystkie pola zostały uzupełnione a rok wydania filmu jest liczbą całkowitą.
- „Delete” – pozwala on na usuwanie pierwszego wybranego rekordu z tabeli.
- „Modify” – otwiera on nowe okno z wcześniejszymi danymi filmu umożliwiając ich modyfikację, jak w przypadku okna otwieranego przyciskiem „Add” sprawdzane są parametry wprowadzane przez użytkownika.
- „Load” – pozwala na załadowanie bazy danych z wybranego pliku „.json”.
- „Save” – pozwala na zapisanie bazy danych do nowego pliku lub nadpisanie porzedniego w formacie „.json”.
- „Refresh” – pozwala na ponowne załadowanie tabeli po wcześniejszym sortowaniu lub wyszukiwaniu.
- „Sort” – pozwala na przesortowanie bazy danych na podstawie wybranej kolumny.
- „Search” – pozwala na wyszukanie dopasowanych wyników do treści wyszukiwania.

3. Analiza problemu

Współczesny użytkownik często korzysta z różnych źródeł informacji i narzędzi do organizacji swojej listy filmów – od prostych notatek, poprzez arkusze kalkulacyjne, aż po dedykowane aplikacje mobilne. Jednak wiele z tych rozwiązań ma ograniczenia, takie jak brak intuicyjnego interfejsu, brak możliwości łatwego udostępniania danych czy brak integracji funkcji takich jak edycja, sortowanie i wyszukiwanie.

Głównym problemem jest brak kompleksowego narzędzia, które umożliwiłoby przechowywanie, organizowanie i przetwarzanie informacji o filmach w sposób przejrzysty i intuicyjny. Użytkownicy często napotykają trudności w:

1. **Zarządzaniu listami filmów** – brak możliwości łatwego dodawania, edytowania czy usuwania danych.
2. **Personalizacji list** – nie zawsze istnieje możliwość tworzenia różnych kategorii, takich jak „Ulubione Filmy” czy „Filmy do Obejrzenia”.
3. **Udostępnianiu danych** – trudność w eksporcie list do pliku, który można przesłać innym użytkownikom.
4. **Przyjazności interfejsu** – brak przejrzystego układu i intuicyjnych funkcji w istniejących rozwiązaniach.

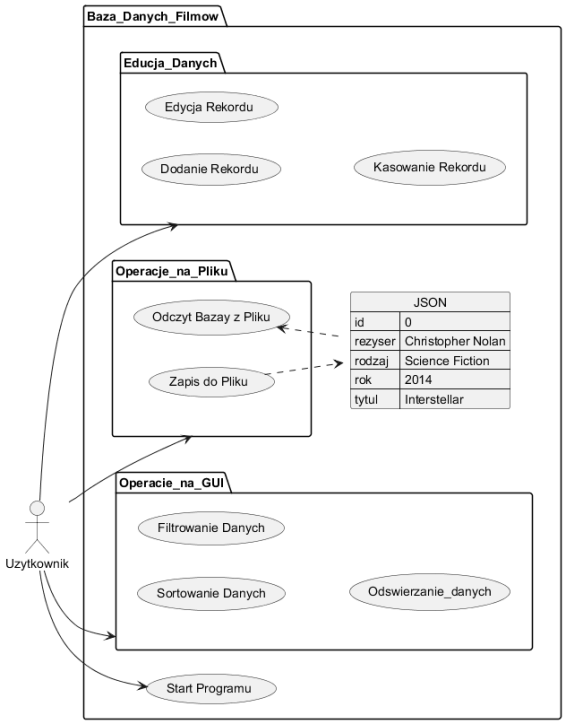
Dodatkowym wyzwaniem jest zapewnienie trwałości i przenośności danych. Wiele narzędzi nie oferuje funkcji zapisu danych do zewnętrznych plików, co utrudnia migrację danych między urządzeniami lub ich archiwizację.

Cele projektu

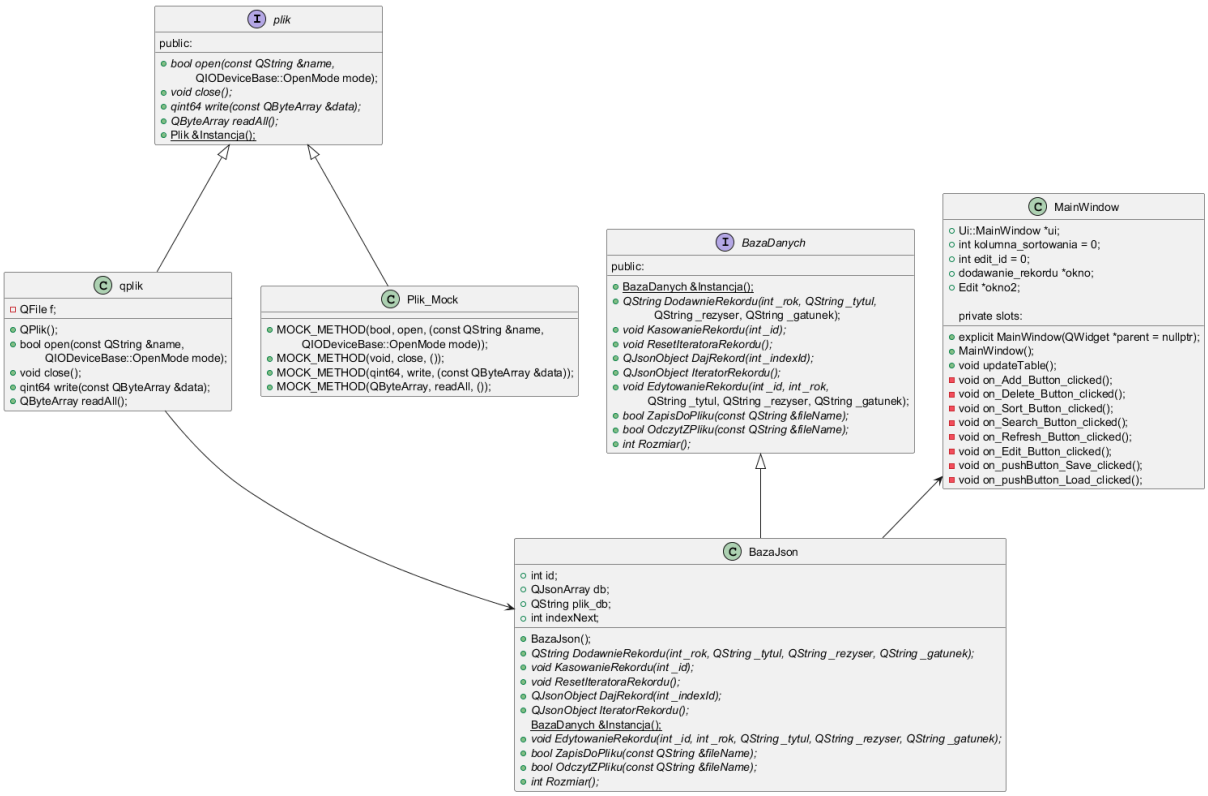
Projektowana aplikacja ma za zadanie rozwiązać powyższe problemy poprzez:

- **Prostą i intuicyjną obsługę** – przejrzysty interfejs z funkcjami wyszukiwania, sortowania i zarządzania rekordami.
- **Personalizację list** – możliwość tworzenia różnych baz filmów i zarządzania nimi.
- **Funkcjonalność zapisu i odczytu** – eksport danych do pliku i import danych z pliku w formacie JSON.
- **Wieloplatformowość** – możliwość skompilowania aplikacji na różnych systemach operacyjnych dzięki wykorzystaniu bibliotek Qt.

4. Projekt techniczny



Rys. 4.1 Przypadek użycia [10]



Rys. 4.2 Diagram klas [11]

5. Opis realizacji

Aby zapewnić odpowiednią jakość i stabilność projektu, wykorzystano dedykowaną platformę testową, która obejmuje sprzęt, oprogramowanie, kompilatory, narzędzia do zarządzania kodem źródłowym oraz systemy kopii zapasowych.

Maszyna testowa:

- Procesor – AMD Ryzen 5 3500X
- Pamięć RAM – 16 GB
- Dysk SSD
- Karta graficzna – Gigabit GeForce GTX 1660
- System operacyjny – Windows 10

Narzędzia:

- Qt Creator – jako główne środowisko IDE używane do tworzenia i debugowania projektu.
- Visual Studio 2022 (opcjonalnie) – alternatywa dla Qt Creator, przetestowanie działania i możliwości budowania projektu.
- Git – jako system kontroli wersji, repozytorium zostało umieszczone na GitHubie.

Kompilatory:

- **MinGW 13.1.0 64-bit** – kompilator GCC używany do budowy projektu w środowisku Windows.
- **MSVC 2022 64-bit** – alternatywny kompilator zgodny z Visual Studio.

6. Opis wykonanych testów

Unit testy zostały zaimplementowane zgodnie z poradnikiem GoogleTest [9].

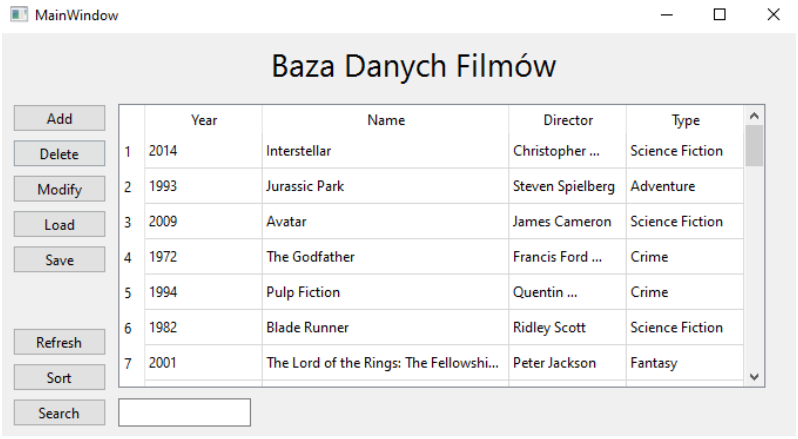
```
[==========] Running 9 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 9 tests from DBTEST
[ RUN      ] DBTEST.test1
[       OK ] DBTEST.test1 (0 ms)
[ RUN      ] DBTEST.otwieranie_nieistnejacego_pliku_do_odczytu
[       OK ] DBTEST.otwieranie_nieistnejacego_pliku_do_odczytu (0 ms)
[ RUN      ] DBTEST.otwieranie_pustego_pliku_do_odczytu
[       OK ] DBTEST.otwieranie_pustego_pliku_do_odczytu (0 ms)
[ RUN      ] DBTEST.otwieranie_pliku_do_odczytu_z_jednym_rekordem
[       OK ] DBTEST.otwieranie_pliku_do_odczytu_z_jednym_rekordem (0 ms)
[ RUN      ] DBTEST.otwieranie_pliku_do_odczytu_z_jednym_elementem_tablicy
[       OK ] DBTEST.otwieranie_pliku_do_odczytu_z_jednym_elementem_tablicy (0 ms)
[ RUN      ] DBTEST.otwieranie_pliku_do_odczytu_nie_json
[       OK ] DBTEST.otwieranie_pliku_do_odczytu_nie_json (0 ms)
[ RUN      ] DBTEST.nieudane_otwarcie_pliku_do_zapisu
[       OK ] DBTEST.nieudane_otwarcie_pliku_do_zapisu (0 ms)
[ RUN      ] DBTEST.udany_zapis_do_pliku
[       OK ] DBTEST.udany_zapis_do_pliku (0 ms)
[ RUN      ] DBTEST.nieudany_zapis_do_pliku
[       OK ] DBTEST.nieudany_zapis_do_pliku (0 ms)
[----------] 9 tests from DBTEST (0 ms total)

[----------] Global test environment tear-down
[==========] 9 tests from 1 test suite ran. (0 ms total)
[  PASSED  ] 9 tests.
```

Manualne testy na GUI

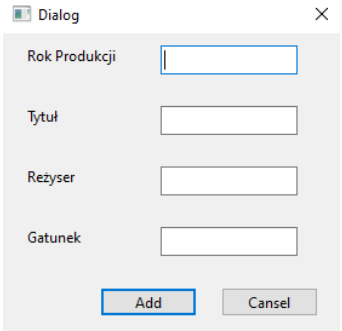
Nazwa Testu	Opis	Autor	Stan
Dodawanie Rekordów	Dodawanie rekordów po przyciśnięciu przycisku „Add”.	Wojciech Midura	PASSED
Usuwanie Rekordu	Usuwanie rekordów po przyciśnięciu przycisku „Delete”.	Wojciech Midura	PASSED
Modyfikacja Rekordu	Zmiana parametrów wskazanego rekordu po wciśnięciu przycisku „Modify”.	Wojciech Midura	PASSED
Ładowanie Bazy Danych z Pliku	Wczytanie istniejącej bazy danych z pliku „database.json” przyciskiem „Load”.	Wojciech Midura	PASSED
Zapisywanie Bazy Danych do Pliku	Zapis bazy danych do nowego pliku lub nadpisanie poprzedniego „database.json” przyciskiem „Save”.	Wojciech Midura	PASSED
Sortowanie Rekordów na Podstawie Kolumny	Zaznaczenie elementu kolumny i wciśniecie przycisku „Sort”.	Wojciech Midura	PASSED
Wyszukiwanie Rekordów w Bazie Danych	Wpisanie w okienko sortowania wybranej danej i wciśnięcie przycisku „Search”.	Wojciech Midura	PASSED
Odświeżanie Tabeli	Po wyszukaniu pojawiają się rekordy z wyniku wyszukiwania aby odświeżyć tabele należy wcisnąć przycisk „Refresh”.	Wojciech Midura	PASSED

7. Podręcznik użytkownika



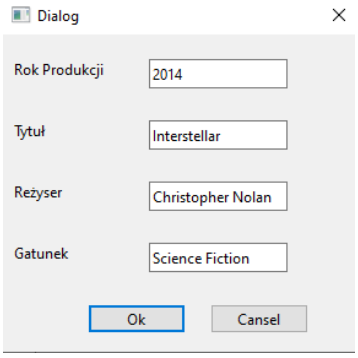
Rys. 7.1 Główne okno bazy danych.

W głównym oknie bazy widoczna jest seria przycisków odpowiedzialnych za nawigacje po bazie danych. Ich działanie zostało opisane w funkcjonalnościach. Widoczne są też kolumny w których są wyświetlane dane rekordów.



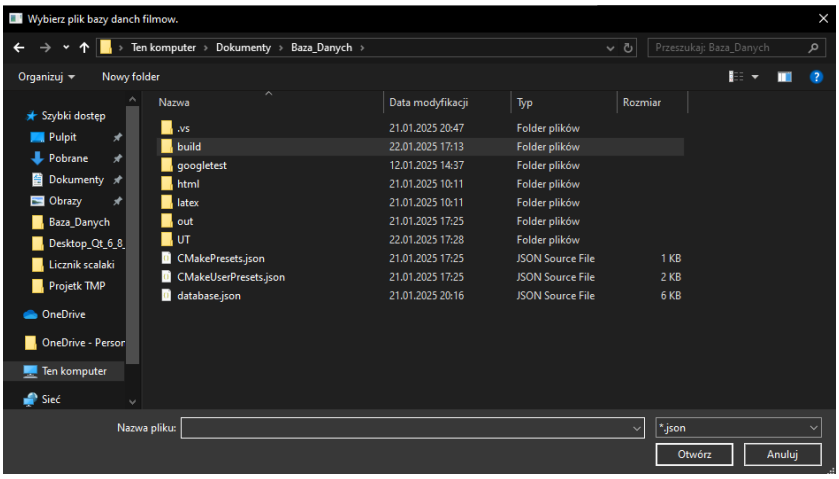
Rys. 7.2 Okno dialogu odpowiedzialnego za dodawanie rekordów.

Okno to wyświetlane jest po naciśnięciu przycisku „Add” w głównym oknie. Ważne jest aby wszystkie rubryki były wypełnione a „Rok Produkcji” był liczbą całkowitą.



Rys. 7.3 Okno dialogu odpowiedzialnego za modyfikacje rekordów.

Okno to wyświetlane jest po naciśnięciu przycisku „Modify” w głównym oknie. Obsługuje te same błędy co okno dodawana ale dodatkowo wyświetlane są parametry modyfikowanego rekordu ułatwiając wprowadzenie zmian.

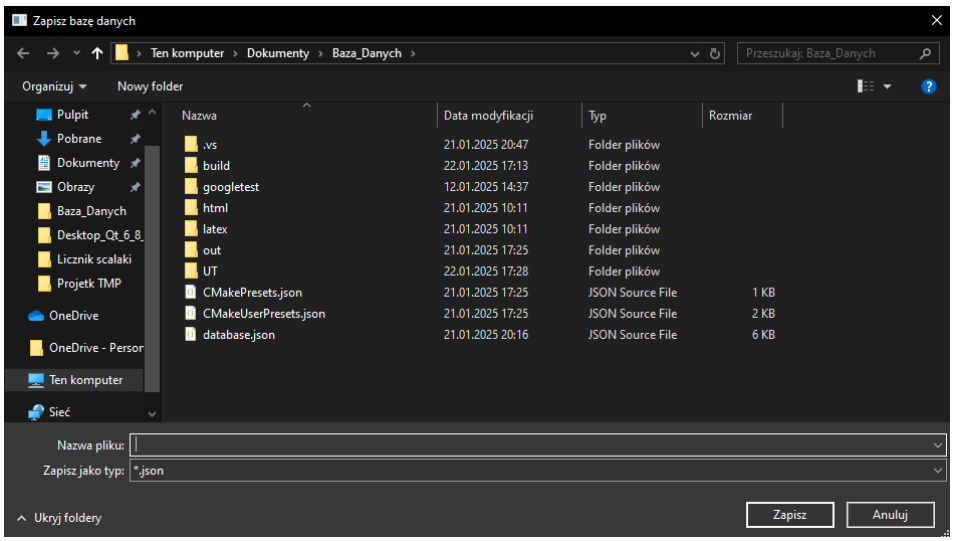


Rys. 7.4 Okno wczytywania bazy danych.

Okno to wyświetlane jest po naciśnięciu przycisku „Load” w głównym oknie. Pozwala ono przeszukiwać pliki w komputerze aby wybrać interesujący nas plik.

Rys. 7.5 Format zapisu rekordu.

Rekordy oprócz tego że muszą być zapisywane w formacie „.json” muszą mieć odpowiednie parametry aby zostały wyświetlone w tabeli (tak jak na Rys. 7.5).



Rys. 7.6 Okno zapisu bazy danych.

Okno to wyświetlane jest po naciśnięciu przycisku „Save” w głównym oknie. Pozwala ono zapisywać plik bazy danych w wybranym przez siebie miejscu na komputerze, umożliwia stworzenie nowego pliku lub nadpisanie poprzedniego w formacie „.json”.

8. Metodologia rozwoju

Pierwszym krokiem po wyborze tematu projektu było ustalenie, jakich narzędzi należy użyć. Wybór padł na narzędzia Qt, które w prosty i przystępny sposób umożliwiają tworzenie interfejsu graficznego. Do realizacji projektu wykorzystano środowisko Qt Creator, choć możliwe jest także zaimportowanie bibliotek jako rozszerzenie do Visual Studio 2022.

Kolejnym istotnym aspektem było zaprojektowanie bazy danych oraz ustalenie formatu, w jakim będą przechowywane dane. Zdecydowałem się na wybór formatu JSOM jako pliku bazodanowego. Dzięki klasie QJsonArray która w prosty sposób pozwala na konwersję danych na plik tekstowy w formacie JSON przy użyciu klasy QJsonDocument. Ważnym elementem realizacji projektu było odizolowanie operacji na bazie danych od interfejsu graficznego, co umożliwia wykonywanie niezależnych testów. Interfejs graficzny korzysta wyłącznie z metod interfejsu BazaDanych, pod którym może istnieć dowolna klasa rzeczywista dziedzicząca po tym interfejsie.

Interfejs graficzny był rozwijany etapowo. Początkowo, oprócz tabeli, znajdowały się w nim jedynie dwa przyciski: „Add” i „Delete”, które umożliwiały dodawanie oraz usuwanie rekordów z bazy danych. W kolejnym etapie dodano przyciski „Sort”, „Search”, „Refresh” oraz „Modify”, które znacząco poprawiły nawigację po rekordach. Przycisk „Refresh” pozwala odświeżyć dane w tabeli, szczególnie po zakończeniu wyszukiwania. Edycję już istniejących danych umożliwia przycisk „Modify”.

Istotnym aspektem realizacji projektu było przypisanie unikalnego identyfikatora (ID) każdemu rekordowi w bazie danych. Dzięki temu usuwanie i modyfikowanie danych odbywa się w sposób jednoznaczny i bezpieczny – raz usunięty rekord z konkretnym ID nie może zostać ponownie utworzony z tym samym identyfikatorem.

W końcowej fazie projektu dodano przyciski „Load” i „Save” oraz ich funkcjonalności. Pozwalają one na zapis wszystkich danych do pliku w formacie „.json” oraz ich odczyt, co umożliwia łatwe zarządzanie różnymi bazami danych i przenoszenie ich między urządzeniami, a także przeglądanie tekstowe.

9. Bibliografia

[1] https://doc.qt.io/qt-6/qjsondocument.html

[2] https://doc.qt.io/qt-6/qjsonarray.html

[3] https://doc.qt.io/qt-6/qjsonobject.html

[4] https://doc.qt.io/qt-6/qfiledialog.html

[5] https://doc.qt.io/qt-6/qtablewidget.html

[6] https://doc.qt.io/qt-6/qmessagebox.html

[7] https://doc.qt.io/qt-6/qfile.html

[8] https://www.youtube.com/watch?v=rH2Kq2BIGVs&t=840s

[9] https://google.github.io/googletest/

[10] https://plantuml.com/use-case-diagram

[11] https://plantuml.com/class-diagram